Mestrado em Engenharia Informática
Departamento de Comunicação e Artes da Universidade de Aveiro

Relatório TDW MP2A

04/12/2023

Nuno Cunha - 98124

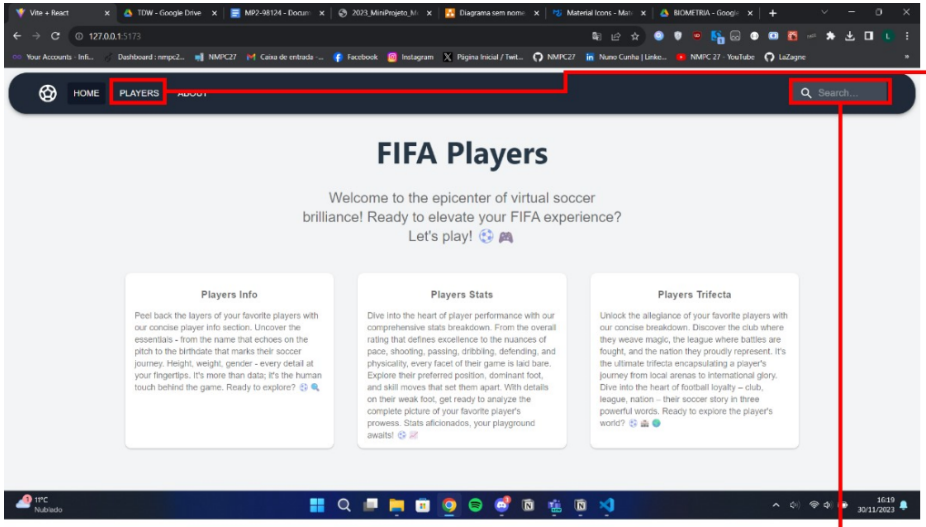
Apresentação do desafio

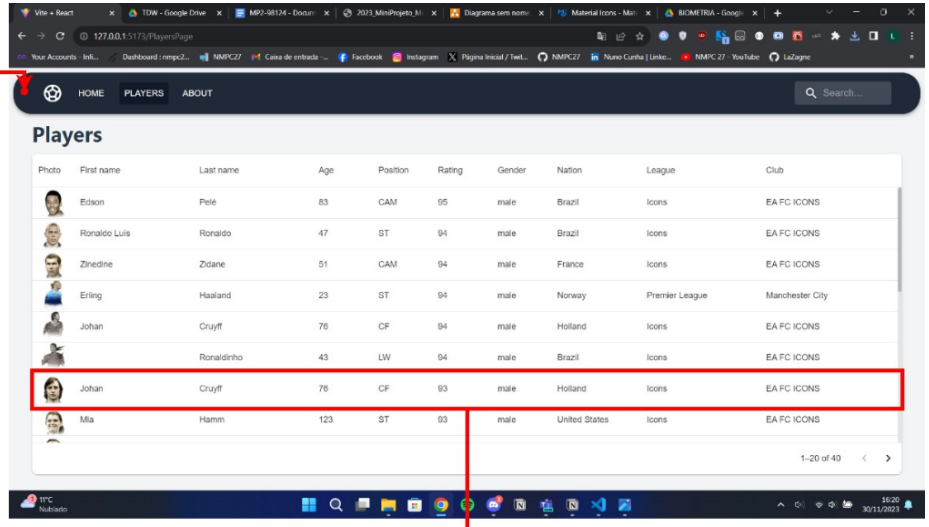
O principal objetivo deste projeto era aproveitar o poder do React, uma biblioteca JavaScript de ponta para a construção de interfaces de utilizador, em conjunto com uma API para desenvolver uma aplicação Web rica em funcionalidades. O foco foi a criação de uma plataforma intuitiva que permitisse aos utilizadores explorar informações detalhadas sobre os jogadores da FIFA, incluindo as suas estatísticas, informações básicas e imagens.

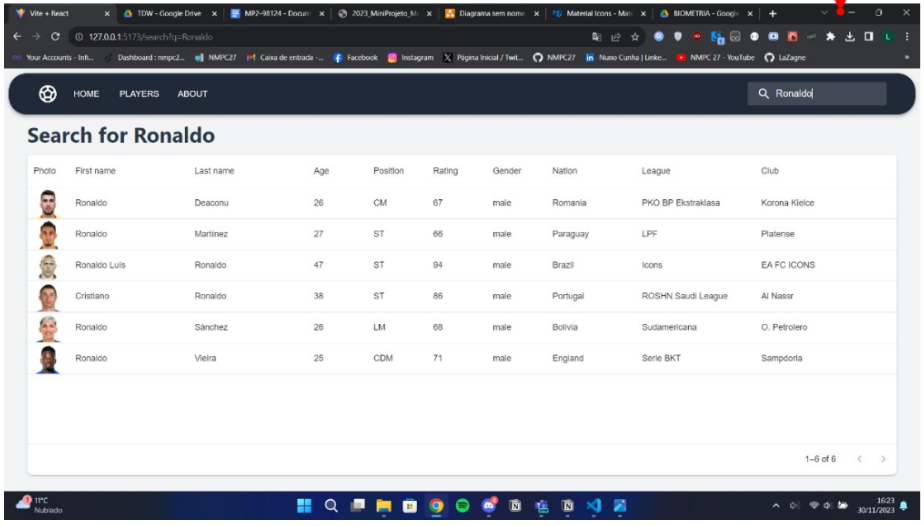
O tema dos jogadores FIFA foi escolhido não só pela sua grande atração entre os entusiastas dos jogos, mas também como prova da versatilidade do desenvolvimento web. Ao combinar a proficiência técnica com a paixão pelos jogos, este projeto procura fazer a ponte entre a tecnologia e o entretenimento.

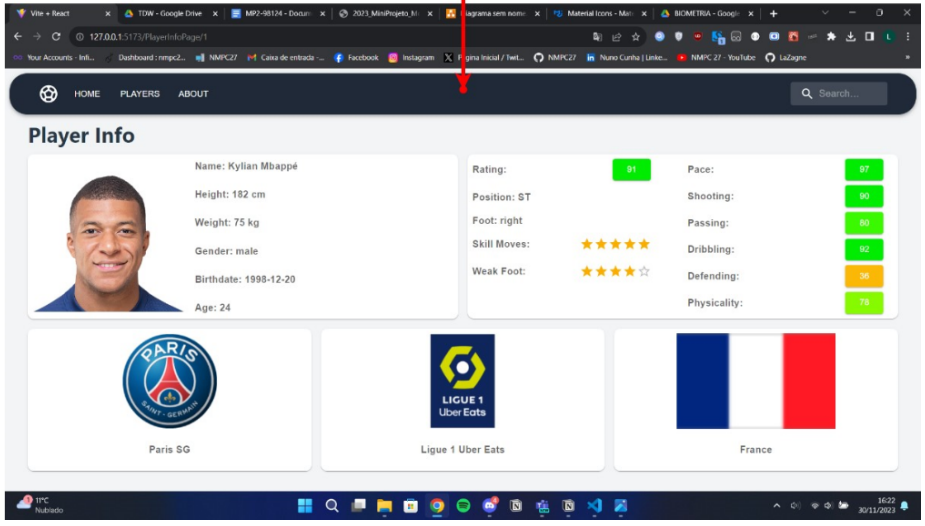
Nas secções que se seguem, vou guiá-lo através do processo de desenvolvimento, mostrando os desafios enfrentados, as soluções implementadas e a evolução geral da aplicação Web FIFA Players.

Design e estrutura da aplicação









As imagens apresentadas acima mostram o design e a estrutura das principais páginas da web app.

Para navegar entre estas páginas é usada uma navbar com links para as principais páginas, home, players e about, e também possui uma search bar para pesquisar o jogador pretendido.

Na home page é uma página “estática” para informar o que o utilizador pode fazer e retirar da web app.

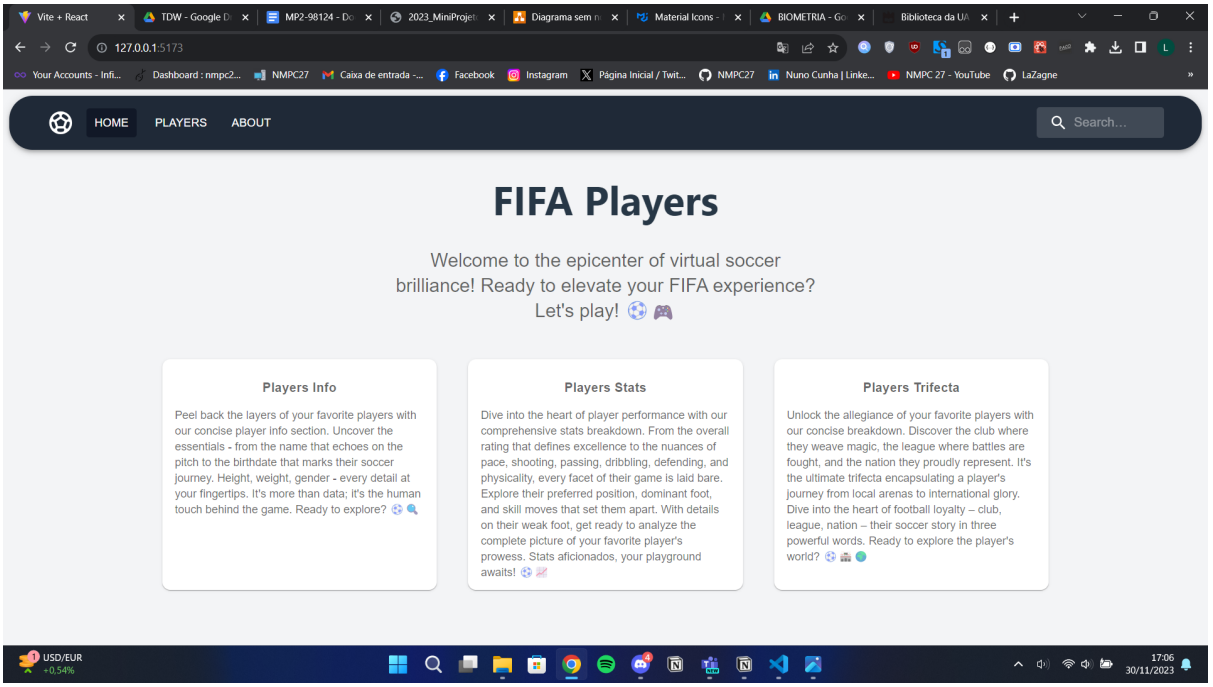
Na players page temos uma tabela com todos os jogadores (mais de 18.000 jogadores), que à medida que vamos trocando de página vai carregando novos jogadores. Na tabela todas as colunas são ordenáveis, mas apenas ordena os jogadores que já carregou, pois a API não suporta qualquer tipo de ordenação. Por último, ao clicar numa coluna o utilizador vai ser redirecionado para a player info page do respectivo jogador.

Ao fazer uma pesquisa na search bar (mínimo de 3 letras por pesquisa) iremos ser redirecionados para a search page onde irão aparecer os jogadores obtidos na pesquisa, na tabela onde estes são apresentados também podemos ordenar todos os campos.

Na player info page é onde são mostradas todas as informações do jogador selecionado anteriormente, esta página funciona de uma maneira “hibrida”, caso os dados não estejam presentes no estado global (redux) então será feito um pedido à API.

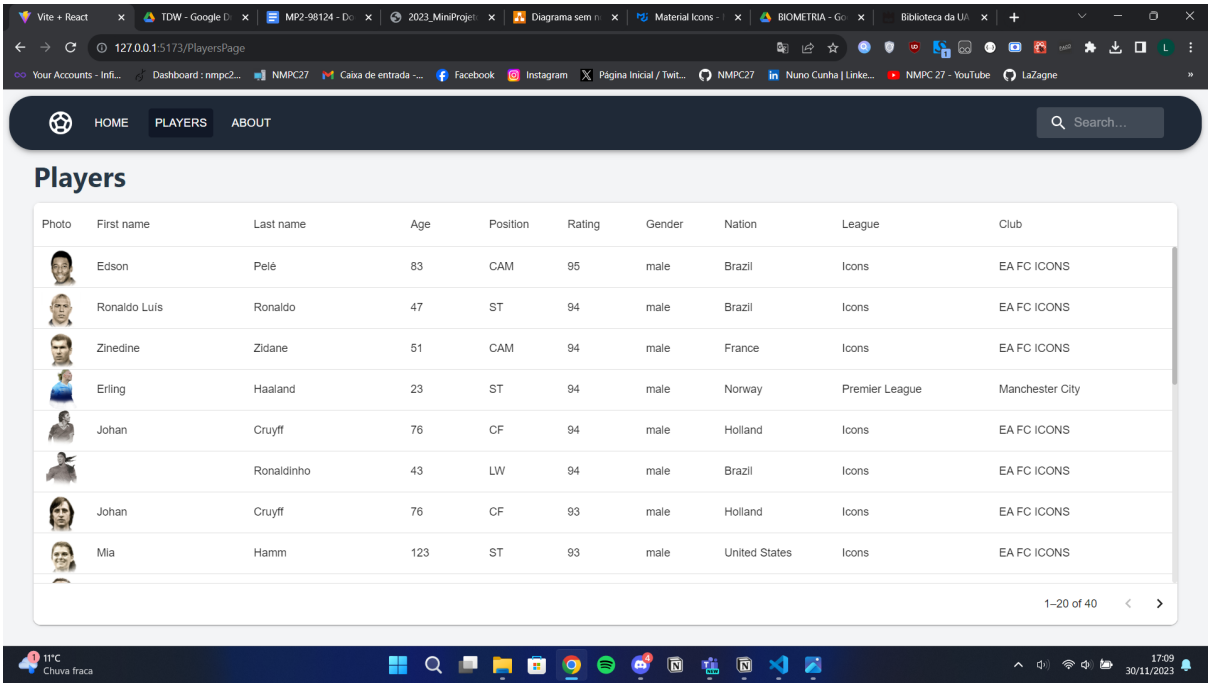
Implementação técnica

Home Page



A home page é uma página “estática” para informar o que o utilizador pode fazer e retirar da web app. Esta página é muito simples por isso considerei que não fosse necessário estar a criar nenhum componente para esta página, o único componente utilizado é a navbar.

Players Page



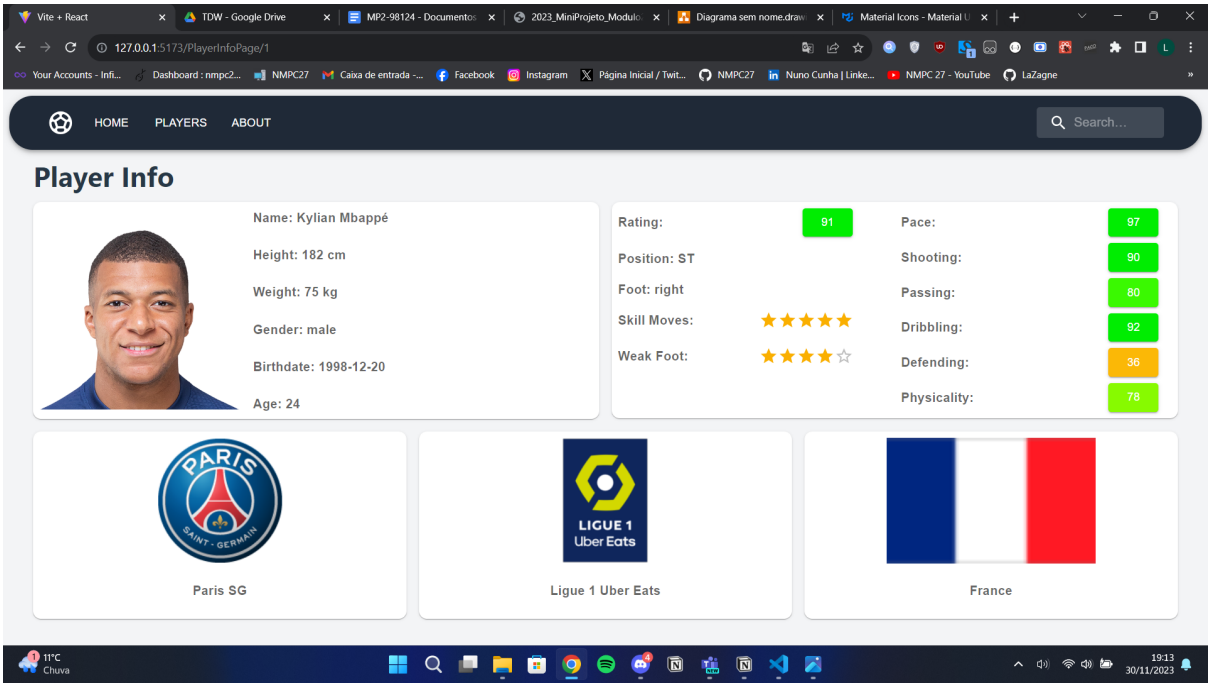
Na players page temos uma tabela com todos os jogadores (mais de 18.000 jogadores), que à medida que vamos trocando de página vai carregando novos jogadores.
Na tabela todas as colunas são ordenáveis, mas apenas ordena os jogadores que já carregou, pois a API não suporta qualquer tipo de ordenação.
Por último, ao clicar numa coluna o utilizador vai ser redirecionado para a player info page do respetivo jogador.

Nesta pagina são apresentados os seguintes dados sobre os jogadores:

- foto
- primeiro nome
- ultimo nome
- idade
- posição
- classificação
- gênero
- nação
- liga
- clube

Estes dados são obtidos usando o Redux-Thunk. No slice (playerSlice.jsx) é chamada a função getPlayersPage(page) que depois faz um pedido ao respectivo endpoint.
Após estes dados serem recebidos os nomes das nações, ligas, clubes e imagens dos jogadores são obtidos através dos ids correspondentes usando as funções getClubByID(id), getLeagueByID(id), getNationsByID(id), getPlayerImgByID(id).

Player Info Page



Na player info page é onde são mostradas todas as informações do jogador selecionado anteriormente, esta página funciona de uma maneira “híbrida”, caso os dados não estejam presentes no estado global (redux) então será feito um pedido à API (para as imagens este pedido é sempre feito).

Esta página tem 2 componentes:

No primeiro componente são mostradas as informações básicas do jogador:

- Foto do jogador
- Nome do jogador
- Altura
- Peso
- Gênero
- Data de nascimento
- Idade

No segundo componente são mostradas as stats do jogador:

- Classificação geral
- Posição
- Pé preferido
- Movimentos de habilidade (estrelas)
- Pé fraco (estrelas)
- Ritmo
- Remate
- Passe
- Drible
- Defesa
- Físico

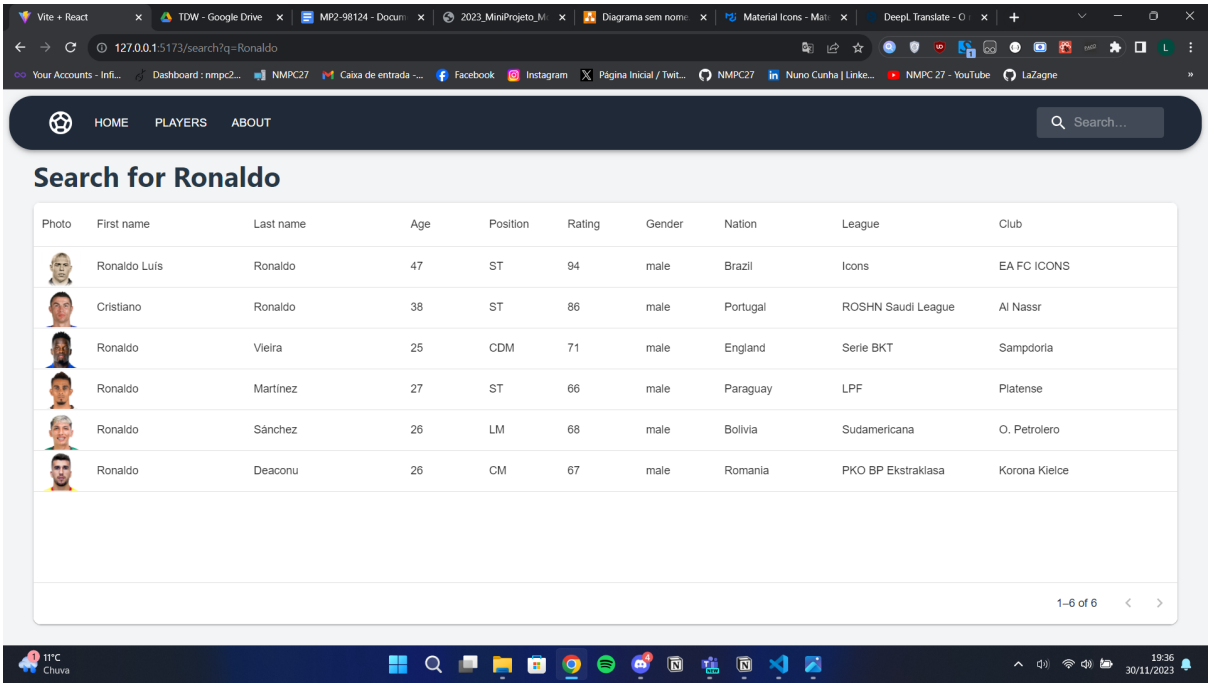
Neste segundo componente é de notar que as cores dos cubos mudam consoante a classificação (gradiente vermelho para verde).

Por último também são apresentadas:

- Logo do clube
- Nome do clube
- Logo da liga
- Nome da liga
- Bandeira da nação
- Nome da nação

Estes dados são mostrados usando o estado global (redux), quando este estado não existe é feito um novo pedido à API o playerID. As únicas exceções a estas regras são as informações do clube, liga, nação e a foto do jogador onde nestes casos o pedido é sempre feito. Mais uma vez estes pedidos são feitos usando Redux-Thunk.

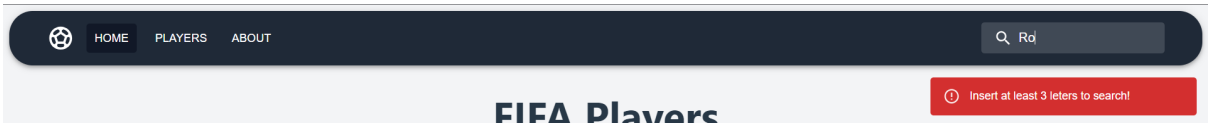
Search Page



Ao fazer uma pesquisa na search bar (mínimo de 3 letras por pesquisa) iremos ser redirecionados para a search page onde irão aparecer os jogadores obtidos na pesquisa, na tabela onde estes são apresentados também podemos ordenar todos os campos.

Esta página é muito semelhante à dos players, por isso tudo o que se aplica à página dos players também se aplica aqui, simplesmente há uma filtração dos dados.

Search by name

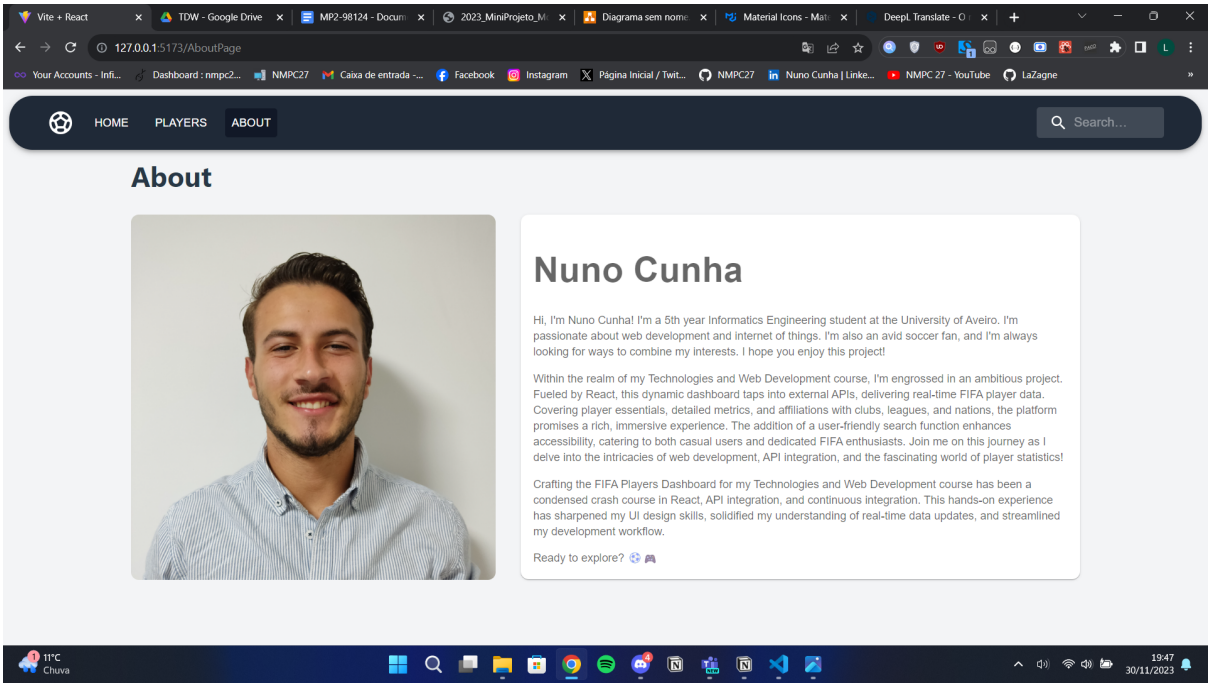


Como a funcionalidade da pesquisa na API é paga, o que eu fiz foi criar um ficheiro (./data/SearchByName.jsx) com um dicionário onde as keys era o nome e os values os repetitivos ids com todos os jogadores (mais de 18.000).

Depois quando o utilizador pesquisa por algo vou a este dicionário encontrar nomes semelhantes, depois faço o request à API com os ids e apresento as informações obtidas na tabela.

Na imagem acima vemos o erro que obriga o utilizador a por pelo menos 3 letras, para não ter de fazer tantos requests à API (limite de 5.000 requests) e para não demorar muito tempo a encontrar os resultados no dicionário.

About Page



Esta página é uma página “estática” onde apenas mostra o autor do projeto e algumas informações sobre este e o próprio projeto.

Navbar

É um componente bastante simples que é usado para navegar entre as páginas, é usado um estado global (Redux) para saber a página atual e alterar os botões consoante isto (ver navbarSlice.jsx).

API

A API usada foi a FutDB, os endpoints e documentação podem ser consultados aqui https://futdb.app/api/doc .

CI

Foi implementado um pipeline de integração contínua onde são corridos o Lint e o Prettier, sempre que era feito um push para a main.

```yaml
name: CI
on:
  push:
    branches:
      - "*"
jobs:
  CI:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repository code
        uses: actions/checkout@v4

      - name: Install Dependencies
        run: npm install

      - name: Run ESLinter
        run: npm run lint

      - name: Run Prettier
        run: npm run prettier
```

Extras

Foram implementadas verificações de Prop-Types sempre que possível.

```
SearchResults.propTypes = {
  playersIDs: PropTypes.object.isRequired,
};
```

Obstáculos

- O número de requests era limitado a 5000 por conta, o que levou à criação de imensas contas.
- A api utilizada devolvia imagens em binário, o que demorou algum tempo a descobrir como se renderiza.

Conclusões

Este projeto melhorou significativamente as minhas competências técnicas, enfatizando a importância de criar um design centrado no utilizador e de aproveitar os dados em tempo real.

Ao longo do processo de desenvolvimento, os desafios encontrados, desde a gestão de estados em React até à estruturação eficaz de componentes, proporcionaram lições inestimáveis em matéria de desenvolvimento Web. Esta experiência prática reforçou a minha proficiência e confiança na resolução de tarefas complexas no ecossistema React.

A integração de APIs no projeto sublinhou as capacidades dinâmicas que estas oferecem às aplicações Web. A obtenção de dados em tempo real enriqueceu a experiência do utilizador, abrindo possibilidades para projectos futuros com uma natureza mais dinâmica e reactiva.

Uma das principais conclusões deste projeto é a ênfase na experiência do utilizador. Encontrar um equilíbrio entre o apelo estético e a intuitividade funcional foi crucial para garantir que os utilizadores pudessem navegar sem esforço na aplicação Web FIFA Players. Esse foco no design thinking acrescenta uma dimensão valiosa ao processo geral de desenvolvimento.

Links

Github: https://github.com/TDW-2023/TDW-M2-cunha/
Nuno Cunha - 98124